Porting / Compiling Applications and Running Jobs on Argonne BlueGene/P

Vitali Morozov and APEDA Team

Argonne NATIONAL LABORATORY

... for a brighter future

U.S. Department of Energy

UChicago Argonne LLC

Office of Science U.S. DEPARTMENT OF ENERGY

A U.S. Department of Energy laboratory managed by UChicago Argonne, LLC

Overview

- Argonne BlueGene/P environment
- Compiling Codes
- Running Jobs
- Improving Performance
- Profiling / Debugging Tools
- Most commonly asked questions

The macro-architecture of the BlueGene/P

is very similar to that of the BlueGene/L,

except that about everything in the system

is

faster and bigger

Configuration Details

- Login Servers
  - compile and submit jobs to Cobalt
  - surveyor.alcf.anl.gov – 13.9T 1-rack BG/P system - testing and development, in production mode
  - endeavour.alcf.anl.gov – 8-rack BG/P system - open for early INCITE users
  - intrepid.alcf.anl.gov – 32-rack BG/P system - acceptance is in progress
- Service Nodes
  - users have restricted access
  - jobs are started from here
  - executable and working directory must be accessible
- I/O Nodes
  - 1/64 IO nodes / compute nodes ratio
  - each compute node is mapped to particular IO node
- Compute Nodes [1024 nodes per rack]
  - users have no access
- Storage Services
  - users have no access

I/O on BlueGene/P

- Home directory
  - GPFS
  - /gpfs/home/<username> -> /home/<username>
  - visible from login, compute, I/O, and service nodes
  - limited in space
  - daily snapshots in ~/.snapshots

- Data
  - PVFS
  - /pvfs-surveyor
  - visible from login, I/O, and compute nodes
  - invisible from the service nodes, so, cannot contain exec, stdin, and stdout files
  - scratch data space, no backups

Building Executable: MPI-Wrapper

- MPI wrappers to IBM compiler set

mpixlc	mpixlcxx	mpixlf77	mpixlf90	mpixlf2003

- Thread-safe versions of MPI wrappers to IBM compiler set

mpixlc_r	mpixlcxx_r	mpixlf77_r	mpixlf90_r	mpixlf2003_r

- MPI wrappers to GNU compiler set

mpicc	mpicxx	mpif77

- BlueGene/L users: change your scripts

mpicc.ibm -> mpixlc	mpicxx.ibm -> mpicxx	mpif77.ibm -> mpixlf77
mpicc.gnu -> mpicc	mpicxx.gnu -> mpicxx	mpif77.gnu -> mpif77

Building Executable: Direct Compiler

/usr/bin/bgcc -> /opt/ibmcmp/vacpp/bg/9.0/bin/bgcc

bgxlc, bgxlc_r	compile C source file
bgxlc++, bgxlc++_r, bgxlC, bgxlC_r	compile C++ source file
bgcc, bgcc_r	compile pre-ANSI C non-standard source file
bgc89, bgc89_r	compile C89-conformed C source file
bgc99, bgc99_r	compile C99-conformed C source file
bgxlf, bgxlf_r, bgf77, bgfort77	compile Fortran 77 source file
bgxlf90, bgxlf90_r, bgf90	compile Fortran 90 source file
bgxlf95, bgxlf95_r, bgf95	compile Fortran 95 source file
bgxlf2003, bgxlf2003_r, bgf2003	compile Fortran 2003 source file

```
DRIVER_PATH=/bgsys/drivers/ppcfloor
bgxlC -o MPI_Prog MPI_Prog.C  -I$DRIVER_PATH/comm/include/ \
   -L$DRIVER_PATH/comm/lib/ -lcxxmpich.cnk -lmpich.cnk -ldcmfcoll.cnk \
   -ldcmf.cnk -lpthread -lrt  -L$DRIVER_PATH/runtime/SPI -lSPI.cna
```

OpenMP Implementation

- Shared-memory parallelism is supported on single node
- Interoperability with MPI as
  - MPI at outer level, across compute nodes
  - OpenMP at inner level, within a compute node
- Thread-safe compiler version should be used
  - with any threaded/OMP/SMP applications
- OpenMP 2.5 standard directives are supported:
  - parallel, for, parallel for, sections, parallel sections, critical, single
  - #pragma omp <rest of pragma> for C/C++
  - !$OMP <rest of directive> for Fortran
- Compiler functions
  - omp_get_num_procs, omp_get_num_threads
    omp_get_thread_num, omp_set_num_threads

BlueGene/P Partitions

- Minimal partition size is 64 nodes: due to one I/O 64 compute node ratio
- Larger partitions are configured by combining smaller ones
- If a job is running on a partition, no other job can run on the enclosing larger partitions
- Not all partitions are available at all times
- bg-listblocks --all lists all defined partitions

Resource Manager and Job Scheduler

- Cobalt – ANL developed job scheduler
- Standard command to manage your jobs

qsub:	submit a job	qstat:	query a job status
qdel:	delete a job	qalter:	alter batched job parameters

- Different queues on surveyor
  - short: 30 minutes, any partition size, run in both midplanes
  - medium: 6pm-6am CST, 512 or 1024-node partitions
  - default: 1 hour jobs only
- FIFO based scheduler
  - chooses the best fit from the top of the queue
- Rack test Day: Tuesday, 9am-5pm CST upon request
- Maintenance Day: Monday, 8am-8pm CST
- Reservations: For special needs send e-mail to

  support@alcf.anl.gov with

  purpose, machine, partition size, start time, duration, special needs

qsub: Submitting a Job

Type qsub

Usage: qsub [-d] [-v] -A <project name> -q <queue> --cwd <working directory>
--env envvar1=value1:envvar2=value2 --kernel <kernel profile>
-K <kernel options> -O <outputprefix> -t time <in minutes>
-e <error file path> -o <output file path> -i <input file path>
-n <number of nodes> -h --proccount <processor count>
--mode <mode> <command> <args>

-t <time_in_minutes>	required runtime
-n <number_on_nodes>	number of nodes
--proccount <number_of_cores>	number of CPUs
--mode <smp|dual|vn>	running mode
--env VAR1=1:VAR2=1	environment variables
<command> <args>	command with arguments

Do not give a partition: it is chosen by a scheduler
If fit to a sooner-to-schedule, a queue is adjusted automatically

qsub: Examples of Submitting a Job

- Despite being redundant, we recommend to always specify the number of nodes, the number of CPUs, and the mode of your run

- qsub -q short -t 10 -n 64 --proccount 64 --mode smp Hello
  - submits a job to a short queue
  - will run no longer than 10 minutes or when executable stops
  - will use smp-mode with 64 nodes, 64 CPUs

- qsub -q short -t 10 -n 4 --proccount 16 --mode vn -O My_Run My_Exe My_File
  - submits a job to a short queue and run no longer than 10 minutes
  - will use vn-mode with 4 nodes, 16 CPUs
  - will allocate 64-node partition, 60 nodes will stay unused
  - will run program My_Exe with argument My_File
  - will create My_Run.output as stdout and My_Run.error as stderr files

qsub: A Script to Submit a Typical Job

```
#!/bin/bash

RUN=<program_executable>
NODES=64
CORES=256
MODE=vn
MAPPING=XYZT

TASK=$RUN-$NODES-$CORES-$MODE

rm -rf $TASK.error $TASK.output

echo Processors: nodes $NODES, cores $CORES, mode $MODE

qsub -q short -t 0:10:00 -n $NODES --proccount $CORES --mode $MODE -O $TASK \
   --env BG_MAPPING=$MAPPING $RUN

qstat -f
touch $TASK.error
tail -f $TASK.error
```

qstat: Show Status of a Batch Job(s)

- qstat -f <job_id1> <job_id2>
  - a full display is produced

```
JobID    JobName     User    WallTime QueuedTime RunTime Nodes State   Location     Mode Procs Queue StartTime
=====================================================================================================
11543 fl-64-64-smp morozov 00:30:00 00:00:06 00:13:41 64 running ANL-R00-M1-N02-64 SMP 64 short 02/27/08
```

  - job_id can be used to kill the job of alter the job parameters
  - valid status: queued, running
  - check the mode of your job

- qstat -Q
  - will show all available queues and their limits
  - special queues, which we use to handle reservations

qdel: Kill a Job

- qdel <jobid1> <jobid2>
  - delete the job from a queue
  - terminated a running job

qalter: Alter Parameters of a Job

- Allows to alter the parameters of both queued and running jobs
- Very useful for the running jobs, which would unexpectedly coming to exceed their allocated time

- Type qalter

Usage: qalter [-d] [-v] -A <project name> -t <time in minutes>

-e <error file path> -o <output file path>

-n <number of nodes> -h --proccount <processor count>

-M <email address> --mode <mode smp/dual/vn> <jobid1> <jobid2>

- Careful: -t <time in minutes>:
  - it is NOT the time left for the running jobs!
  - it is elapsed time since the beginning of the run, after which Cobalt kills the job

Why a job is not running in a queue

- there is a reservation, which interferes with your job
  - showres shows all reservations currently in place
- there is no available partitions
  - partlist shows all partitions marked as functional
  - partlist shows the assignment of each partition to a queue
- wrong queue
  - the job submitted to a queue, which is restricted to run at this time
- partitions are not freed
  - in specific situations, a job quits and does not free a partition => a partition is treated as busy, but there is no job, which holds this partition
  - bg-listblocks --all --long prints full information of all blocks
  - the state is identified by a combination of qstat -f, bg-listblocks

Tools: Improved Performance, Profiling, Debugging …

- Most tools are under /soft/apps
- Improved performance with optimized libraries
  - BLAS/LAPACK versus LibGOTO/LAPACK
  - BlueGene optimized Mass, MassV, ESSL libraries from IBM
- Practical Optimization
  - compiler switches
  - profiling and profiling tools: HPCT, Profiling “-pg”, “-qdebug=function_trace”, TAU
- Tracing MPI_Barrier/printf/exit/abort standard debugging methods
- GDB / Totalview
  - the last choice, requires to perform additional arrangements, reservation, and specific step-by-step instructions
  - need close work with support team

Optimization Steps w/o Code Changes

- Start from original MPI program, make it run
  - The least aggressive compiler options
  - Default libraries
- Increase compiler optimization options
- Verify different running modes: smp vs. dual vs. vn
- Use highly optimized libraries (BLAS-LibGOTO, MASSV, ESSL)
- Optimize communication performance: DCMF_EAGER
- Optimize mapping (logical MPI-task to CPU allocation): BG_MAPPING
- Use compiler directives
  - Alignment, aliasing, loop unrolling, SIMD vectorization

Optimization Steps with Code Changes

- Profiling (identify the bottleneck)
  - Profiling Tools with and without code modification
  - Use of hardware counters
  - Start code changes only if the bottleneck is concentrated
- Rearranging memory hierarchy
  - Ordered memory inquires improve cache reuse (Fortran N-dim arrays)
  - Use of contiguous memory blocks allows quadword loads
- Use double-hummer instructions
  - Available for Fortran, C, C++ as regular calls
  - Register/instructions scheduler is done by compiler
- Last choice: hand-coding assembly
  - Assembly generated by a compiler is a great help to understand the code

Memory Hierarchy

- L1 Instruction and L1 Data caches
  - 32 KB total size, 32-Byte line size, 64-way associative, round-robin
  - -qcache=level=1:type=d:assoc=64:line=32:size=32:\
    level=1:type=i=assoc=64:line=32:size=32
- L2 Data cache
  - 2KB prefetch buffer, 16 lines, 128-byte a line
  - -qcache=level=2:type=c:line=128:size=2
- L3 Data cache
  - 8 MB, 35 cycles latency
  - -qcache=level=3:type=c:line=128:size=8192:cost=35
- Memory size
  - 2GB DDR-2 at 400 MHz, 86 cycles
- Memory bandwidth
  - in L1-cache: ffpdx/stfpdx instructions, 1 quadword load/cycle: 16B*850 /s = 13.6 GB/s
  - out of L1-cache: complex memory hierarchy

IBM XL Compiler General Optimization

- Default: -qarch=[450|450d] -qnoautoconfig -qstaticlink -qtune=450
- -O0: no optimization, implies -qstrict_induction (no loop counter optim)
- -O = -O2: balanced optimization, implies -qstrict_induction -qstrict
- -O3 -qstrict: preserves program semantics
- -O3 = -O2 -qfloat=fltint:rsqrt:norngchk -qmaxmem=1 -qhot=level=1: aggressive but reasonably stable level
- -qhot: turns on High-Order loop analysis and Transformation unit
  - arraypad, level, simd, vector
- -qreport: produces a listing, shows how code was optimized
- -qipa: interprocedural analysis, use with caution
  - level, inline, list

IBM XL Compiler BG-Specific Optimization

- Architecture flags
  - -qalign: Fortran only, specifies the alignment of data
  - -qarch=450: generates PPC450 instructions
  - -qarch=450d: generates double-hummer instructions
- Increase of optimization aggressiveness
  - -O -qarch=450: default optimization level
  - -O3 -qarch=450/450d
  - -O4 -qarch=450d -qtune=450
  - -O4 = -O3 -qarch -qtune -qcache -qhot -qipa=level=1
  - -O5 = -O4 -qipa=level=2
- -qlistopt: generates the listing with all flags used in compilation

Example program

```
#define SIZE 1024

double A[SIZE][SIZE];
double B[SIZE][SIZE];
double C[SIZE][SIZE];

double multiply(void)
{
  int i, j, k;

  for (i = 0; i < SIZE; i ++)
    for (j = 0; j < SIZE; j++)
      for (k = 0; k < SIZE; k++)
       C[i][j]
         += A[i][k] * B[k][j];

  return C[SIZE-10][SIZE-10];
}
```

-qreport: shows, how sections of code have been optimized

```
do {
   /* id=3 guarded */ /* ~10 */
   /* region = 52 */
   /* bump-normalized */
   /* independent */
   $.CSE15 = $.ICM0 + $.CIV3;
   $.CSE17 = B[$.ICM3][$.CSE15];
   $.CSE16 = C[$.ICM6][$.CSE15] + $.ICM7 * $.CSE17;
   C[$.ICM6][$.CSE15] = $.CSE16;
   $.CSE18 = B[$.ICM8][$.CSE15];
   C[$.ICM6][$.CSE15] = $.CSE16 + $.ICM9 * $.CSE18;
   $.CSE19 = C[$.ICMA][$.CSE15] + $.ICMB * $.CSE17;
   C[$.ICMA][$.CSE15] = $.CSE19;
   C[$.ICMA][$.CSE15] = $.CSE19 + $.ICMC * $.CSE18;
   $.CIV3 = $.CIV3 + 1;
} while ((unsigned) $.CIV3 < (unsigned) $.ICME);
```

Example program

- -qsource: produces a listing with source section
- -qlist: produces an object listing

```
lfpdx    fp4,fp36=B[]0(gr21,gr29,0,offset=8)
addi     gr21=gr21,32
lfpdx    fp3,fp35=B[]0(gr21,gr24,0,offset=-8)
fxcpmadd fp1,fp33=fp7,fp39,fp0,fp32,fp10,fp10,fcr
fxcpmadd fp6,fp38=fp9,fp41,fp0,fp32,fp11,fp11,fcr
fxcpmadd fp0,fp32=fp5,fp37,fp4,fp36,fp12,fp12,fcr
fxcpmadd fp4,fp36=fp2,fp34,fp4,fp36,fp13,fp13,fcr
fxcpmadd fp2,fp34=fp1,fp33,fp3,fp35,fp12,fp12,fcr
fxcpmadd fp1,fp33=fp6,fp38,fp3,fp35,fp13,fp13,fcr
stfpdx   C[]0(gr22,gr30,0,offset=-8184)=fp0,fp32
stfpdx   C[]0(gr22,gr29,0,offset=8)=fp4,fp36
```

Runtime Mode

- SMP mode
  - qsub --mode smp
  - Single MPI task on CPU0 / 2 GB RAM
- Dual mode
  - qsub --mode dual
  - Two MPI tasks on a node / 1GB RAM each
- Virtual Node mode
  - qsub --mode vn
  - Four MPI tasks on a node / 512 MB RAM each

Threading Support

- OpenMP is supported
  - NPTL pthreads implementation in glibc requires NO modifications

- Compute Note Kernel supports
  - execution of one quad-threaded process

    (each of the CPUs is assigned to each of maximum 4 threads)
  - execution of two two-threaded processes
  - execution of four single-threaded processes
  - proper mode should be specified for qsub

MPI Mapping

- Default XYZT mapping
  - (XYZ) are torus coordinates, T is a CPU number
  - X-coordinate is increasing first, then Y, then Z
  - All XYZT permutations are possible
- qsub --env BG_MAPPING=TXYZ --mode vn …
  - This puts MPI task 0,1,2,3 to Node 0 CPU0, CPU1, CPU2, CPU3; MPI tasks 4,5,6, and 7 to Node2 CPU0,CPU1,CPU2,CPU3
  - Typically, default XYZT is less efficient than TXYZ mapping
- qsub --BG_MAPPING=<FileName> --mode smp …
  - use high-performance toolkits to determine communication pattern
  - optimize mapping by custom mapfile
  - mapfile: each line contains 4 coordinates to place the task, first line for task 0, second line for task 1…
  - avoid conflict in mapfiles (no verification)

Optimized libraries

- BG-optimized BLAS Level 1,2,3 library from Kazushige Goto, U. of Texas
- IBM ESSL library: BLAS1, 2, 3 in /soft/apps/ESSL
- Generic versions of BLAS/LAPACK/FFTW

-O0	102.82 s	20.88 MFlop/s
-O2	70.86 s	30.31 MFlop/s
-O3	3.471 s	618.52 MFlop/s
-O4	7.921 s	271.10 MFlop/s
-O5	7.919 s	271.12 MFlop/s
ESSL	0.836 s	2569.6 MFlop/s 75.76 % of peak
GOTO	0.828 s	2593.0 MFlop/s 76.26 % of peak

Performance Toolkits

- Compiler options for profile information
  - no instrumentation, simple to use
  - -pg
  - gprof <exe> gmon.out.0
- TAU - Tuning and Analysis Utilities
  - /soft/apps/tau/tau-latest
  - requires additional instrumentation
  - extensive visualization capabilities
  - can be combined with PAPI-3.9.0 hardware counters*
- HPCT - IBM High-Performance Computing Toolkit
  - /soft/apps/hpct_bgp
  - New product, not much feedback is available, esp. for large projects
  - MPI profiling and tracing tool, CPU Profiling, Hardware Counter Performance Monitoring, I/O Performance

Use of gprof Tool with Compiler Options

- Profiling is collecting and arranging statistics of running program
- Simple to use: does NOT require instrumentation of sources
- Use -p option at compile AND link time
- Use -g option, but remember that it removes automatic inlining
- Run program: it will produce gmon.out.N binary files, one for each MPI task
- Convert a binary to readable text format:

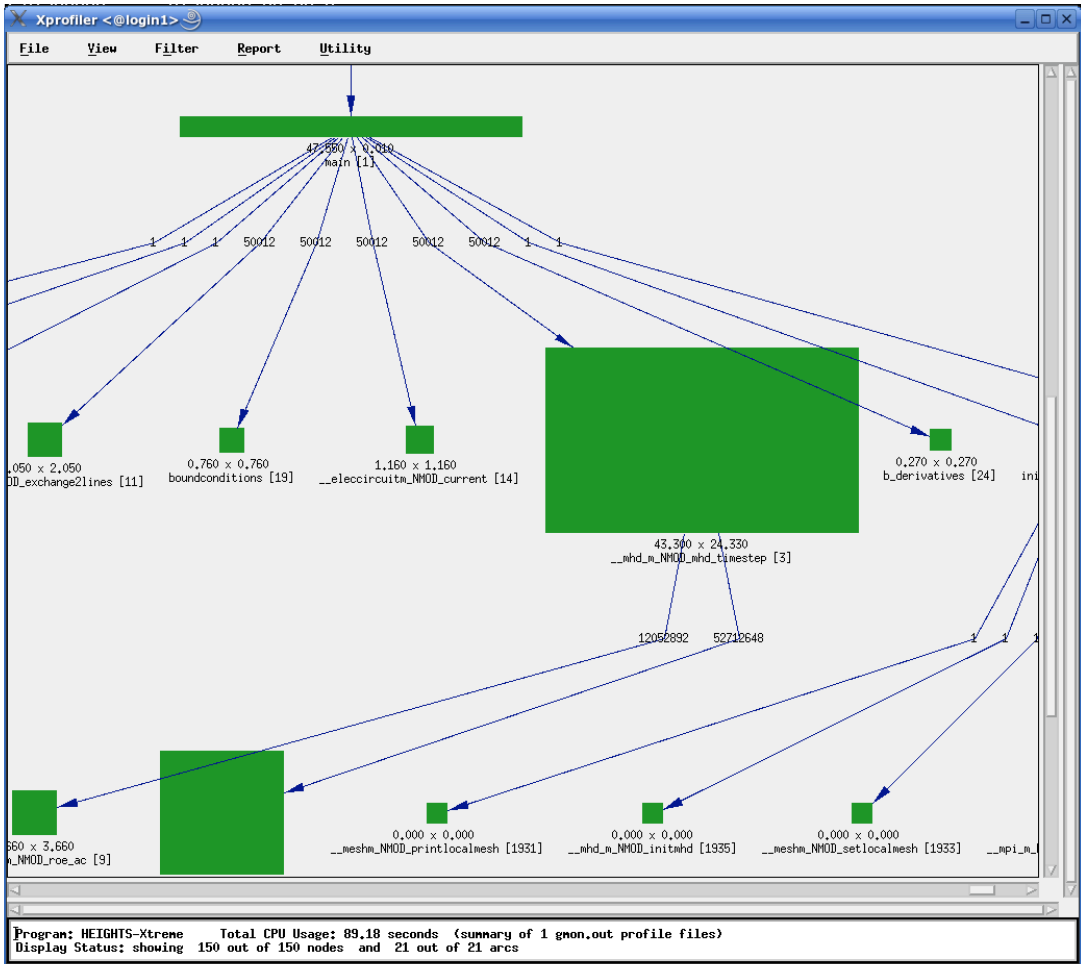
  gprof <executable> gmon.out.0
- Alternatively, use Xprofiler graphical tool (part of HPCT)
- http://www.gnu.org/software/binutils/manual/gprof-2.9.1/gprof.html

Flat profile

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 32.49      24.33    24.33    50012     0.00     0.00  __mhd_m_NMOD_mhd_timestep
 20.45      39.64    15.31 52712648     0.00     0.00  __thermom_NMOD_roe_peta
  7.09      44.95     5.31                             DCMF::hwBarrier::poll()
  7.08      50.25     5.30                             DMA_RecFifoSimplePollNormalFifoById
  4.89      53.91     3.66 12052892     0.00     0.00  __thermom_NMOD_roe_ac
  3.02      56.18     2.27
   DCMF::Queueing::Lockbox::LockboxMessage::advance()
  2.74      58.23     2.05    50012     0.00     0.00  __mpi_m_NMOD_exchange2lines
  2.27      59.93     1.70
   DCMF::Protocol::MultiSend::TreeAllreduceRecvPostMessage::advanceDeep(DCMF::Queueing::Tree::TreeMs
   gContext)
  1.80      61.28     1.35                             DCMF::DMA::Device::advance()
  1.55      62.44     1.16    50012     0.00     0.00  __eleccircuitm_NMOD_current
  1.32      63.42     0.99                             DCMF::Queueing::Lockbox::Device::advance()
  1.23      64.34     0.92                             DCMF_Messager_advance
  0.05      73.54     0.04                             DCMF_Send
  0.05      73.58     0.04                             MPIDI_BG2S_RecvCB
  0.05      73.62     0.04                             DCMF::DMA::Device::processAdvanceQueue()
```

- Search for functions with larger time usage
- Search for functions with larger number of calls

HPCT GUI Tool - Xprofiler



Use of TAU toolkit

- Tuning and Analysis Utilities (TAU): A toolkit for profiling and performance analysis of parallel programs from the University of Oregon
- Requires instrumentation of the source.

  Includes tau_XXX scripts for automatic instrumentation:

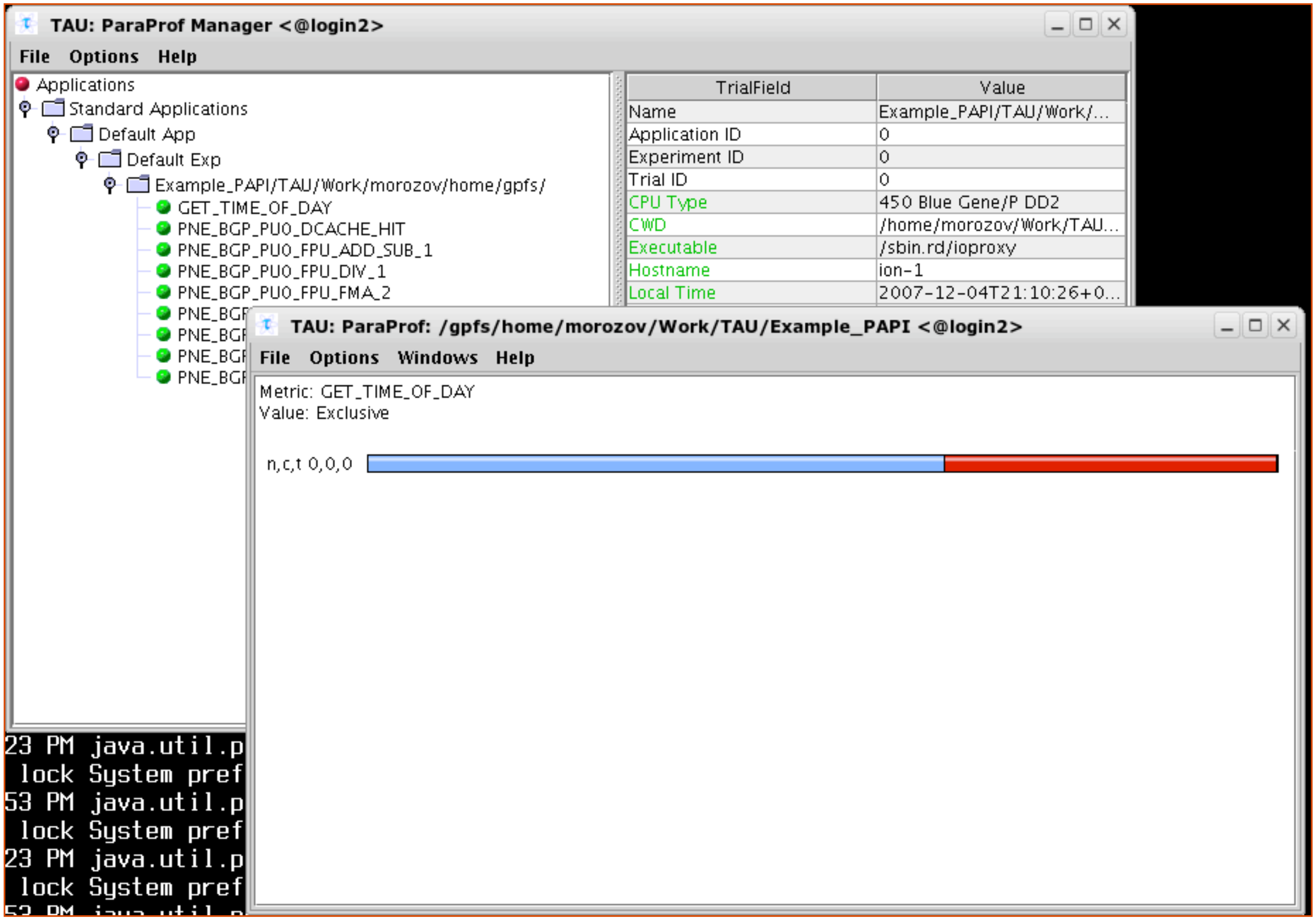
  mpicxx -o computePi computePi.cpp

  changed to

  export TAU_MAKEFILE=/soft/apps/tau/tau_latest/bgp/lib/Makefile.tau-multiplecounters-mpi-papi-pdt

  tau_cxx.sh -o computePi computePi.cpp
- Running MPI program as usual produces profile.NNN files, one for each MPI task
- Using paraprof tool to explore the performance and visualize data
- http://www.cs.uoregon.edu/research/tau/docs/newguide.index.html

TAU GUI Tool - Paraprof



Use of TAU toolkit: Selective Instrumentation

```
#include <Profile/Profile.h>
…
TAU_PROFILE( "main", "int( int,char**)", TAU_DEFAULT);
TAU_PROFILE_SET_NODE(0);
TAU_PROFILE_TIMER( t1, "name", "void(void)", TAU_USER );
…
TAU_PROFILE_START( t1);
/* code to profile */
TAU_PROFILE_STOP( t1 )
```

IBM HPCT Tool for MPI/CPU/IO Profile

- IBM High Performance Computing Toolkit - HPCT
  - Tools to visualize and analyze your performance data
  - Xprofiler and HPCT GUI instructions
  - Tools to optimize your application's performance
- MPI Profiling and Tracing (mpitrace)
- CPU Profiling with -pg and gmon.out.X
- Hardware Counter Performance Monitoring
- I/O Performance
- Located on /soft/apps/hpct_bgp
- http://www.redbooks.ibm.com/abstracts/redp4256.html

Example of using HPCT Tool

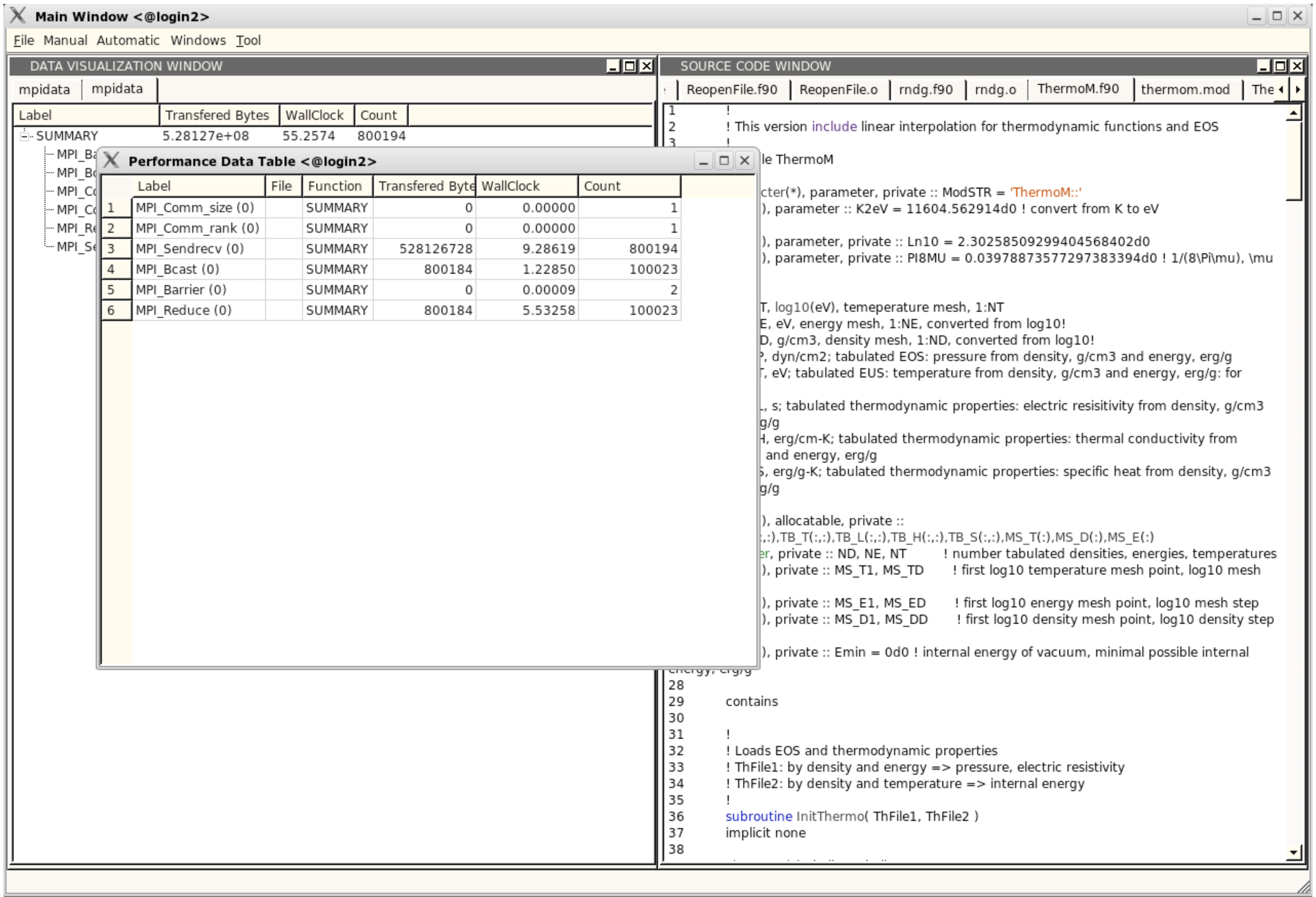
- Instrument the program
- See listing
- Use hardware counters
- Change optimization options

```
#include "libhpm.h"

hpmInit( taskID, "hpct-hcpm" );
hpmStart( 1, "multiply-regular" );
  for (i = 0; i < SIZE; i ++)
      for (j = 0; j < SIZE; j++)
          for (k = 0; k < SIZE; k++)
              C[i][j] += A[i][k] * B[k][j];
hpmStop( 1 );
hpmTerminate( taskID );
```

```
HPCT_DIR=/soft/apps/hpct_bgp
mpixlcxx_r -I$HPCT_DIR/include -o Hello Hello.cxx -L$HPCT_DIR/lib
-lhpm -llicense
```

HPCT GUI Tool - Peekperf

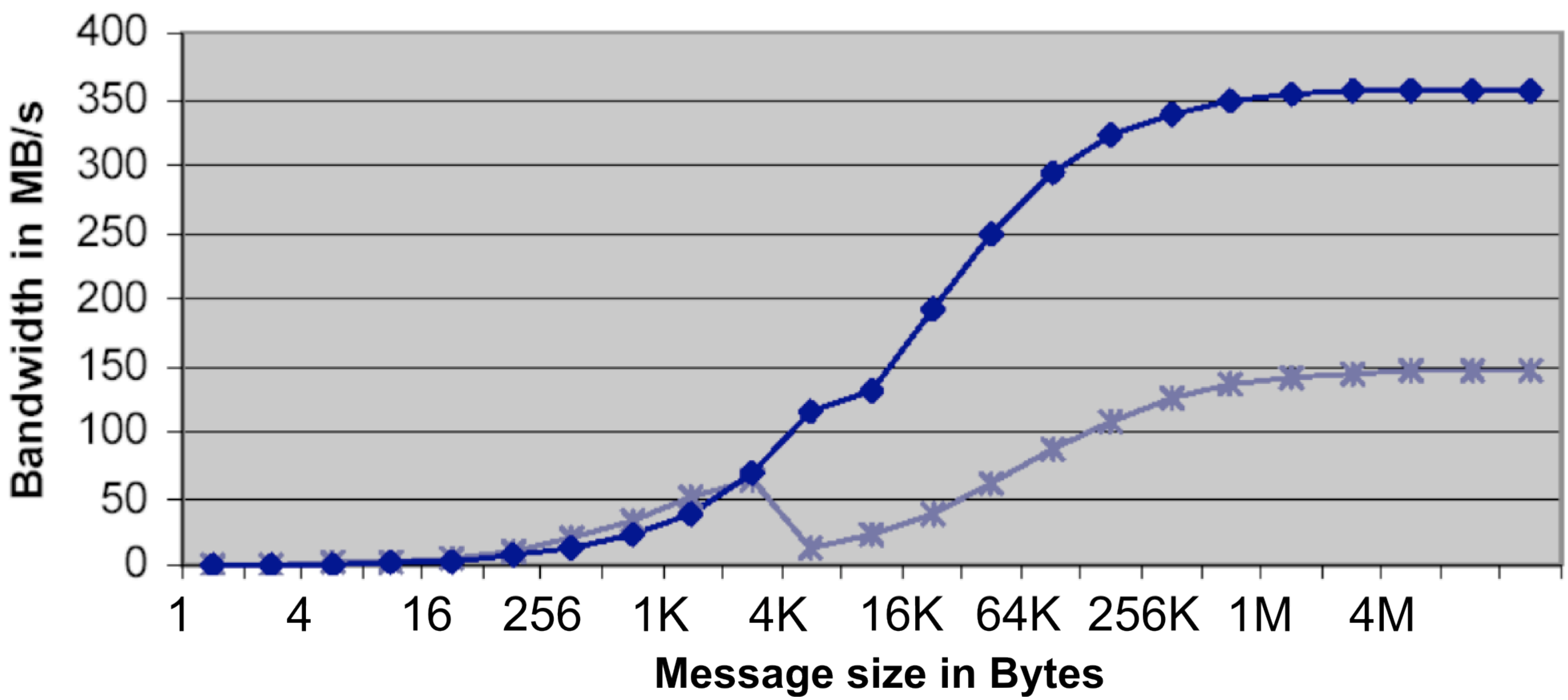


Communication Operations

- Best if no use of complex derived data
- For performance reason, it is advisable do not overlap p2p and collective operations
- P2P operations a figure from Application Development
  - Routing messages statically or dynamically
  - Control routing by DCMF_EAGER variable (changes the rendezvous threshold)
- Collective operations: latency and bandwidth from Application Development
  - Collective operations are more efficient than p2p, and should be used if possible

Point-to-point Operations

- Intel MPI PingPong benchmark: BG/L co-mode vs. BG/P smp-mode
- Nearest neighbor communication
- The break line is due to switching from short to eager



IBM System Blue Gene Solution: Blue Gene/P Application Development RedBook

BlueGene/P Collective Operations

- Intel MPI Collective Benchmark
- Preferred over P2P due to lower overhead, independent on mapping

MPI Routine	Condition	Network	Performance
MPI_Barrier	MPI_COMM_WORLD	barrier (global interrupt) network	1.2 µs
MPI_Barrier	any communicator	torus network	30 µs
MPI_Broadcast	MPI_COMM_WORLD	collective network	817 MB/sec
MPI_Broadcast	rectangular communicator	torus network	934 MB/sec
MPI_Allreduce	MPI_COMM_WORLD fixed-point	collective network	778 MB/sec
MPI_Allreduce	MPI_COMM_WORLD floating point	collective network	98 MB/sec
MPI_Alltoall[v]	any communicator	torus network	84-97% peak
MPI_Allgatherv		torus network	same as broadcast

IBM System Blue Gene Solution: Blue Gene/P Application Development RedBook

Personality* of BlueGene/P

```
#include <common/bgp_personality.h>
#include <common/bgp_personality_inlines.h>

_BGP_Personality_t p;

Kernel_GetPersonality( &p, sizeof(p) );

p.DDR_Config.DDRSizeMB;              /* memory size */
p.Kernel_Config.ProcessConfig;       /* running mode */
p.Network_Config.Xnodes;             /* torus dimensions */
p.Network_Config.Ynodes;
p.Network_Config.Znodes;
```

mpixlc_r -I/bgsys/drivers/ppcfloor/arch/include …

Debugging on BlueGene/P

- GDB
  - mpirun must be used (not Cobalt), and therefore
  - request a reservation through support@alcf.anl.gov
  - use step-by-step instructions
    from /software/common/doc/BGP-Using-gdb.txt

- Totalview
  - will be available shortly

Tuning code for BlueGene/P

- Structuring data in adjacent pairs
  - Allows to use quadword load/store operations
- Using vectorizable blocks
  - Organize the code sequences with single entry point
  - Minimize branching for special cases (exceptions, NaN values)
  - Minimize dependencies between blocks
- Minimize the usage of C/C++ pointers, guarantee disjoint references
- Use inline (with caution)
  - to remove overhead with brunching
  - to enlarge the vectorizable blocks
- Turn off range checking -qfloat=norngchk (with caution)

Tuning code for BlueGene/P (cont.)

- Removing possibilities for aliasing
  - Reduces overhead from reloading data
  - Local variables allows better use of registers
- Structuring floating-point computations
  - PPC450 pipelines fp-operations with latency of several cycles and a result of each cycle afterwards
  - Automatic or manual loop unrolling helps to fill-up the conveyer
- Checking for data alignment
  - makes use of load/store quadword instructions
  - load/store values should not cross a 32-bytes cache-line boundary

Questions and Answers: Compiling I

Q **My program does not compile or link; there are undefined symbols or definitions that seem to be from system libraries?**

A Make sure you are:

1. cross-compiling with correct mpi-wrapper script or a bg-prefixed compiler;
2. not using /usr/include include files;
3. not using and /usr/lib library files.

System libraries are located in /bgsys/drivers/ppcfloor.

Application libraries and tools are located in /soft/apps

Questions and Answers: Compiling II

Q **I am getting undefined references at link time:
undefined reference to `_xlf_create_threadlocal'**

A Use a thread-safe version of a compiler, which has an _r suffix (mpixlf90_r instead of mpixlf90 in this example).

Questions and Answers: Compiling III

Q **I am getting the errors from mpicxx.h declaration file at compile time:**

```
#error directive:
"SEEK_SET is #defined but must not be for the C++ binding of MPI"
"SEEK_CUR is #defined but must not be for the C++ binding of MPI"
"SEEK_END is #defined but must not be for the C++ binding of MPI"
```

A Put #include <mpi.h> before #include <stdio.h> in your C++ source file.

Questions and Answers: Running I

Q **How do I get each line of the output labeled with the MPI rank that it came from?**

A Include "--env MPIRUN_LABEL=1" environment variable in your qsub command.

Questions and Answers: Running II

Q **How can I see the output of my batch job sooner / before it is finished?**

A Short answer:

qsub --env MPIRUN_ENABLE_TTY_REPORTING=0 ... a.out

Long story:

- The output of a job is buffered for performance.
- Standard input/output/error streams of a job are connected to files.
- The output to devices are not buffered.
- Tell Cobalt that standard streams are devices, not files.

Questions and Answers: Running III

Q **My job had empty stdout file. stderr file shows that the job died immediately after it started. What happened?**

<Feb 29 14:31:02.873207> BE_MPI (ERROR): The error message in the job record is as follows:
<Feb 29 14:31:02.873237> BE_MPI (ERROR): "killed with signal 6"

A The executable is too large to load.

Use 'size a.out' command to determine approximate size.

An executable should require:

- less than 2 GB in smp-mode
- less than 1 GB in dual-mode
- less than 512 MB in vn-mode

Resources

- ALCF Resource page
  http://www.alcf.anl.gov/support/usingALCF/index.php

- Getting Started
  http://www.alcf.anl.gov/support/gettingstarted/index.php

- IBM RedBooks:
  Compiler User Guides, Application Development Manuals
  http://www.redbooks.ibm.com/redbooks.nsf/redbooks/

Help!

- Running Your Application Problems
  - Porting, Compiling, and Running your jobs
  - Performance issues, Tuning, Scaling, Debugging, Profiling
  - Unexpected results, core dumps, apparently wrong result

  Consult with your Catalyst or E-mail to support@alcf.anl.gov

- Login problems, access problems, environment problems, Job scheduler problems, system not responding

  E-mail to support@alcf.anl.gov